ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Bragin M.D., Rogov B.V.

Bicompact schemes for multidimensional hyperbolic equations on Cartesian meshes with solution-based AMR

KELDYSH INSTITUTE OF APPLIED MATHEMATICS

Russian Academy of Sciences

M. D. Bragin, B. V. Rogov

Bicompact schemes for multidimensional hyperbolic equations on Cartesian meshes with solution-based AMR

Moscow — 2019

Michael Dmitrievich Bragin, Boris Vadimovich Rogov

Bicompact schemes for multidimensional hyperbolic equations on Cartesian meshes with solution-based AMR

High-order bicompact schemes for hyperbolic equations on Cartesian meshes with solution-based adaptive mesh refinement are constructed. The algorithm for implementation of these schemes on such meshes is described in detail. A new solution-based criteria of mesh refinement is proposed. Bicompact schemes with this refinement criteria are tested on the two-dimensional problem of compactly supported pulse advection and the two-dimensional Sedov blast wave problem. It is shown, that the design of bicompact schemes allows them to be implemented on meshes of such class with good accuracy of the computed solution ensured.

Keywords: bicompact schemes, high-order schemes, hyperbolic equations, adaptive mesh refinement.

Брагин М. Д., Рогов Б. В.

Бикомпактные схемы для многомерных уравнений гиперболического типа на декартовых сетках с адаптацией к решению

Для уравнений гиперболического типа построены высокоточные бикомпактные схемы на декартовых сетках с адаптацией к решению. Подробно описан алгоритм реализации бикомпактных схем на таких сетках. Предложен новый критерий адаптации сетки к решению. Бикомпактные схемы с этим критерием адаптации проверены на двумерной задаче о переносе финитного импульса и двумерной задаче Седова о сильном взрыве в идеальном газе. Показано, что конструкция бикомпактных схем допускает счет на сетках данного класса, обеспечивая при этом хорошую точность вычисляемого решения.

Ключевые слова: бикомпактные схемы, высокоточные схемы, гиперболические уравнения, сетки с адаптивным измельчением.

This research was supported by the Russian Foundation for Basic Research, project no. 18-31-00045.

Introduction

The development of high-order accurate schemes for the numerical solution of partial differential equations, including hyperbolic ones, remains an important area of computational mathematics. The high order of accuracy (three and higher) makes it possible to achieve a prescribed error on coarser meshes, which is especially important for solving multidimensional problems [1]. In some applications, for example, in aeroacoustics [1,2], elasticity theory [3,4], and electrodynamics [5], the evolution of wave perturbations has to be accurately computed at long distances and times, so high-order accurate schemes are highly preferable to low order ones.

A new class of high-order accurate schemes for systems of nonstationary multidimensional quasilinear hyperbolic equations was developed in [6–9]. These schemes involve a compact approximation of spatial derivatives. Since their spatial stencil includes only two integer nodes, the schemes [6–9] are called bicompact.

Bicompact schemes combine several positive properties. They have the fourth order of accuracy in space, which can be increased to the sixth, eighth, and so on (see [10]). At the same time, the difference order of the bicompact schemes' equations in the independent space variables coincides with the order of the partial differential equations to be solved. As a result, the number of boundary conditions in the difference problem coincides with that in the exact formulation of the problem. Bicompact schemes are conservative and make it possible to choose a time stepping method, since they are derived using the finite-volume method and the method of lines. Moreover, bicompact schemes preserve their high order of accuracy in space on highly nonuniform meshes, since the spatial stencil of the schemes occupies a single mesh cell. In the dispersion and dissipation properties, bicompact schemes are superior to many well-known finite-difference and compact schemes [10,11]. Finally, bicompact schemes can be efficiently implemented by applying space marching computation, including its parallel version [12]. This combination of properties makes bicompact schemes advantageous over other high-order accurate schemes.

However, the applicability of bicompact schemes has been limited to computational domains of simple geometry (such as rectangles and parallelepipeds) and to Cartesian meshes. This limitation can be overcome in two ways, namely, by constructing bicompact schemes on unstructured meshes or by applying the well-known well-elaborated approach of Cartesian meshes with adaptive mesh refinement (AMR) [13,14], which has become highly popular in recent time [15–21]. The latter method is more attractive, since, in fact, it is reduced to a more sophisticated implementation of previously known schemes and does not require the construction of new ones.

The generalization of bicompact schemes to Cartesian meshes with AMR (which will also be referred to hereafter as adaptive Cartesian meshes or Cartesian meshes with adaptation) splits into two sequential subproblems: generalization to Cartesian meshes with solution-based AMR and generalization to Cartesian meshes with adaptation to the geometry of the computational domain. The goal of this work is to solve the former subproblem in principle in the case of a simple computational domain. We consider only bicompact schemes of fourth-order accuracy in space, although all arguments and constructions presented below can be carried over to bicompact schemes of higher orders.

This preprint is organized as follows. Bicompact schemes and their implementation on Cartesian meshes without AMR are described in Section 1. In Section 2, we present a numerical algorithm using bicompact schemes on Cartesian meshes with solution-based AMR and propose a new mesh refinement criterion. In Section 3, bicompact schemes on Cartesian meshes with solution-based AMR are tested as applied to the two-dimensional advection of a compactly supported pulse and the two-dimensional Sedov blast wave problem.

1. Bicompact schemes on standard Cartesian meshes

Consider a system of two-dimensional homogeneous quasilinear hyperbolic equations in the simplest computational domain D:

$$\begin{gathered}\mathscr{L}_2(\boldsymbol{Q}) \equiv \partial_t \boldsymbol{Q} + \partial_x \boldsymbol{F}(\boldsymbol{Q}) + \partial_y \boldsymbol{G}(\boldsymbol{Q}) = \boldsymbol{0}, \\ (x, y) \in D = (0, x_{\max}) \times (0, y_{\max}), \quad t \in (0, t_{\max}),\end{gathered} \tag{1}$$

where $\boldsymbol{Q} = (Q_1, \ldots, Q_m) = \boldsymbol{Q}(x, y, t)$ is the sought vector of conservative variables, $\boldsymbol{F}(\boldsymbol{Q})$ and $\boldsymbol{G}(\boldsymbol{Q})$ are the vectors of physical fluxes in the Ox and Oy directions, respectively, and $\partial_x \equiv \partial/\partial x$. Assume that, at $t = 0$ and on the boundary ∂D of D, system (1) is supplemented with initial and boundary conditions under which it has a unique solution in $\overline{D} \times [0, t_{\max}]$, $\overline{D} = D \cup \partial D$.

Before continuing our narration, we explain why two-dimensional equations, rather than one- or three-dimensional ones were chosen for consideration. The two-dimensional case is of interest for two reasons.

1. It is more complicated than the one-dimensional case to a degree sufficient to study all important features and issues concerning the implementation of bicompact schemes on adaptive Cartesian meshes.

2. It is simpler than the three-dimensional case, i. e., free of tedious three-dimensional finite-difference mathematics.

For system (1), we consider the following unsplit (in x, y) baseline bicompact scheme of fourth-order accuracy in x, y [7]:

$$\begin{cases} A_0^y A_0^x \left(\boldsymbol{Q}_C^{n+1} - \boldsymbol{Q}_C^n\right) + \tau A_0^y \Lambda_1^x \boldsymbol{F}_C^{n+1} + \tau \Lambda_1^y A_0^x \boldsymbol{G}_C^{n+1} = \boldsymbol{0}, \\ A_0^y \Lambda_1^x \left(\boldsymbol{Q}_C^{n+1} - \boldsymbol{Q}_C^n\right) + \tau A_0^y \Lambda_2^x \boldsymbol{F}_C^{n+1} + \tau \Lambda_1^y \Lambda_1^x \boldsymbol{G}_C^{n+1} = \boldsymbol{0}, \\ \Lambda_1^y A_0^x \left(\boldsymbol{Q}_C^{n+1} - \boldsymbol{Q}_C^n\right) + \tau \Lambda_1^y \Lambda_1^x \boldsymbol{F}_C^{n+1} + \tau \Lambda_2^y A_0^x \boldsymbol{G}_C^{n+1} = \boldsymbol{0}, \\ \Lambda_1^y \Lambda_1^x \left(\boldsymbol{Q}_C^{n+1} - \boldsymbol{Q}_C^n\right) + \tau \Lambda_1^y \Lambda_2^x \boldsymbol{F}_C^{n+1} + \tau \Lambda_2^y \Lambda_1^x \boldsymbol{G}_C^{n+1} = \boldsymbol{0}. \end{cases} \tag{2}$$

The notation in scheme (2) is explained as follows. In the closed domain $\overline{D}$, we introduce a standard Cartesian mesh $\Omega = \Omega_x \times \Omega_y$ without AMR, where

$$\Omega_x = \{x_0, x_{1/2}, x_1, x_{3/2}, x_2, \ldots, x_{N_x}\}, \quad x_0 = 0, \quad x_{N_x} = x_{\max},$$
$$h_{x,j+1/2} = x_{j+1} - x_j \equiv h_x \text{ — is the step size in } x, \quad x_{j+1/2} = \frac{x_j + x_{j+1}}{2};$$
$$\Omega_y = \{y_0, y_{1/2}, y_1, y_{3/2}, y_2, \ldots, y_{N_y}\}, \quad y_0 = 0, \quad y_{N_y} = y_{\max},$$
$$h_{y,k+1/2} = y_{k+1} - y_k \equiv h_y \text{ — is the step size in } y, \quad y_{k+1/2} = \frac{y_k + y_{k+1}}{2}.$$

The time interval $t \in [0, t_{\max}]$ is divided into a set of levels:

$$\{t^0, t^1, \ldots, t^{N_t}\}, \quad t^0 = 0, \quad t^{N_t} = t_{\max},$$
$$\tau^{n+1/2} = t^{n+1} - t^n \equiv \tau \text{ — is the step size in } t.$$

The letter C denotes the multi-index $(j + 1/2,\ k + 1/2)$, $j = \overline{0, N_x - 1}$, $k = \overline{0, N_y - 1}$, $n = \overline{0, N_t - 1}$. The mesh vector function $\boldsymbol{Q}^n$ approximates the exact solution of system (1) at nodes of the mesh Ω at the time $t = t^n$. For an arbitrary mesh function U, the difference operators $A_0^d, \Lambda_1^d, \Lambda_2^d$, $d \in \{x, y\}$ are defined by the formulas

$$A_0^x U_{j+1/2,k'} = \frac{U_{j,k'} + 4U_{j+1/2,k'} + U_{j+1,k'}}{6}, \quad \Lambda_1^x U_{j+1/2,k'} = \frac{U_{j+1,k'} - U_{j,k'}}{h_x},$$
$$\Lambda_2^x U_{j+1/2,k'} = \frac{4(U_{j,k'} - 2U_{j+1/2,k'} + U_{j+1,k'})}{h_x^2},$$

$$A_0^y U_{j',k+1/2} = \frac{U_{j',k} + 4U_{j',k+1/2} + U_{j',k+1}}{6}, \quad \Lambda_1^y U_{j',k+1/2} = \frac{U_{j',k+1} - U_{j',k}}{h_y},$$
$$\Lambda_2^y U_{j',k+1/2} = \frac{4(U_{j',k} - 2U_{j',k+1/2} + U_{j',k+1})}{h_y^2},$$

where $j' \in \{j,\ j + 1/2,\ j + 1\}$ and $k' \in \{k,\ k + 1/2,\ k + 1\}$. The flux vectors in scheme (2) are computed as $\boldsymbol{F}_{j',k'}^{n+1} = \boldsymbol{F}(\boldsymbol{Q}_{j',k'}^{n+1})$, $\boldsymbol{G}_{j',k'}^{n+1} = \boldsymbol{G}(\boldsymbol{Q}_{j',k'}^{n+1})$.

In addition to the unsplit scheme (2), we are interested in its more computationally efficient version with locally one-dimensional (LOD) splitting in [22–24], see also [25]:

$$\begin{cases} A_0^x\left(\boldsymbol{Q}^{[x]}_{j+1/2,\,k'} - \boldsymbol{Q}^{[x*]}_{j+1/2,\,k'}\right) + \tau\Lambda_1^x\boldsymbol{F}^{[x]}_{j+1/2,\,k'} = \boldsymbol{0}, \\ \Lambda_1^x\left(\boldsymbol{Q}^{[x]}_{j+1/2,\,k'} - \boldsymbol{Q}^{[x*]}_{j+1/2,\,k'}\right) + \tau\Lambda_2^x\boldsymbol{F}^{[x]}_{j+1/2,\,k'} = \boldsymbol{0}; \\ A_0^y\left(\boldsymbol{Q}^{[y]}_{j',\,k+1/2} - \boldsymbol{Q}^{[y*]}_{j',\,k+1/2}\right) + \tau\Lambda_1^y\boldsymbol{G}^{[y]}_{j',\,k+1/2} = \boldsymbol{0}, \\ \Lambda_1^y\left(\boldsymbol{Q}^{[y]}_{j',\,k+1/2} - \boldsymbol{Q}^{[y*]}_{j',\,k+1/2}\right) + \tau\Lambda_2^y\boldsymbol{G}^{[y]}_{j',\,k+1/2} = \boldsymbol{0}. \end{cases} \tag{3}$$

Here, $\boldsymbol{Q}^{[d]}$ and $\boldsymbol{Q}^{[d*]}$, $d \in \{x, y\}$, are auxiliary mesh functions related by the formulas

$$\boldsymbol{Q}^{[x*]} = \boldsymbol{Q}^n, \quad \boldsymbol{Q}^{[y*]} = \boldsymbol{Q}^{[x]}, \quad \boldsymbol{Q}^{n+1} = \boldsymbol{Q}^{[y]}. \tag{4}$$

The flux vectors in scheme (3) are computed as in scheme (2). The indices j', k' in (3) can take not only strictly internal, but also boundary values, depending on the boundary conditions specified for system (1).

LOD splitting can be used variously. Namely, the one-dimensional equations can be solved first in x and then in y, as indicated in (4); on the contrary, they can be solved in y and, then, in x (the letters x and y in the superscripts in (4) are then interchanged); or the resulting solution $\boldsymbol{Q}^{n+1}$ can be found as the half-sum of the solutions obtained using the first two unsymmetrized versions of splitting. The last symmetrized version of the splitting scheme with a half-sum makes it possible to increase the order of accuracy in t up to the second.

The fully discrete bicompact schemes (2) and (3) are derived from corresponding semi-discrete schemes by approximating the time derivatives using an implicit Euler method. Integration of these semi-discrete schemes with the help of higher order DIRK methods yields bicompact schemes of higher order accuracy in t. If the chosen DIRK method has an order $p \geqslant 2$, then the truncation error of the unsplit bicompact scheme is $\boldsymbol{O}(\tau^p, h^4)$, while the error of the bicompact LOD scheme is $\boldsymbol{O}(\tau, h^4)$ for unsymmetrized splitting and $\boldsymbol{O}(\tau^2, h^4)$ for symmetrized splitting, where $h = \max\{h_x, h_y\}$.

It is well known that high-order multistage DIRK methods are implemented by combining implicit Euler methods with suitable stage time steps and input data. In the same manner, bicompact schemes of higher order accuracy in t are implemented by combining baseline bicompact schemes. Therefore, to study the implementation of all bicompact schemes, it is sufficient to analyze schemes (2) and (3).

If the Jacobian matrices $\boldsymbol{A}(\boldsymbol{Q}) = \partial_{\boldsymbol{Q}}\boldsymbol{F}(\boldsymbol{Q})$ and $\boldsymbol{B}(\boldsymbol{Q}) = \partial_{\boldsymbol{Q}}\boldsymbol{G}(\boldsymbol{Q})$ are neither positive nor negative definite for any $\boldsymbol{Q}$ allowed by system (1), then the implementation of schemes(2), (3) requires the Lax–Friedrichs global flux splitting. This procedure makes use of the vectors

$$\boldsymbol{F}^{\pm}(\boldsymbol{Q}) = \frac{1}{2}\boldsymbol{F}(\boldsymbol{Q}) \pm C_2^x\boldsymbol{Q}, \quad \boldsymbol{G}^{\pm}(\boldsymbol{Q}) = \frac{1}{2}\boldsymbol{G}(\boldsymbol{Q}) \pm C_2^y\boldsymbol{Q},$$

where the flux splitting parameters C_2^d, $d \in \{x, y\}$ are chosen so that the Jacobian matrices satisfy the inequalities

$$\partial_{\boldsymbol{Q}}\boldsymbol{F}^{+}(\boldsymbol{Q}),\ \partial_{\boldsymbol{Q}}\boldsymbol{G}^{+}(\boldsymbol{Q}) > 0, \quad \partial_{\boldsymbol{Q}}\boldsymbol{F}^{-}(\boldsymbol{Q}),\ \partial_{\boldsymbol{Q}}\boldsymbol{G}^{-}(\boldsymbol{Q}) < 0$$

for all values of the exact solution of system (1) occurring in $\overline{D}$ in a neighborhood of the considered instant of time. In the transition from the level t^n to t^{n+1}, the flux splitting parameters are computed explicitly using the estimates

$$\begin{aligned} C_2^x &= \frac{1+2\delta}{2}V_{\max}^x, \quad V_{\max}^x = \max_{\substack{s=\overline{1,m}\\(x,y)\in\Omega}} |\lambda_s(\boldsymbol{Q}^n(x,y);\boldsymbol{A})|,\\ C_2^y &= \frac{1+2\delta}{2}V_{\max}^y, \quad V_{\max}^y = \max_{\substack{s=\overline{1,m}\\(x,y)\in\Omega}} |\lambda_s(\boldsymbol{Q}^n(x,y);\boldsymbol{B})|, \end{aligned} \tag{5}$$

where $\lambda_s(\boldsymbol{Q};\boldsymbol{X})$ is the sth eigenvalue of the matrix $\boldsymbol{X}(\boldsymbol{Q})$ and $\delta > 0$ is a "reserve factor of positive/negative definiteness" of the matrices $\partial_{\boldsymbol{Q}}\boldsymbol{F}^{\pm}, \partial_{\boldsymbol{Q}}\boldsymbol{G}^{\pm}$.

Finally, the implementation of schemes (2), (3) can be described as follows. The transition from the level t^n to t^{n+1} is performed in several stages:

1. The parameters C_2^d, $d \in \{x, y\}$ are computed using formula (5).
2. The vectors $\boldsymbol{F}$ and $\boldsymbol{G}$ in (2), (3) are replaced by the corresponding split fluxes $\boldsymbol{F}^{\pm}$, $\boldsymbol{G}^{\pm}$ with some choice of signs. To be definite, let $\boldsymbol{F} = \boldsymbol{F}^{+}$ and $\boldsymbol{G} = \boldsymbol{G}^{+}$; the other choices are considered in a similar manner.
3. The cells $[x_j, x_{j+1}] \times [y_k, y_{k+1}]$ of the mesh Ω are double looped through with respect to j, k, $j = \overline{0, N_x - 1}$, $k = \overline{0, N_y - 1}$. In each cell, the equations of the chosen scheme are solved for the unknown quantities $\boldsymbol{Q}$ at the nodes $(x_{j+1/2}, y_{k+1/2})$, $(x_{j+1/2}, y_{k+1})$, $(x_{j+1}, y_{k+1/2})$, and (x_{j+1}, y_{k+1}). In scheme (2), for example, Newton's method or iterated approximate factorization [26] is used for this purpose. Newton's method is applied in scheme (3). Note that a single loop over the cells involves the computation of only $\boldsymbol{Q}^{[x]}$ (or only $\boldsymbol{Q}^{[y]}$, which depends on the splitting version). The remaining $\boldsymbol{Q}^{[y]}$ (or $\boldsymbol{Q}^{[x]}$) is computed in the other loop.
4. The signs of flux splitting are reversed, so that now $\boldsymbol{F} = \boldsymbol{F}^{-}$ and $\boldsymbol{G} = \boldsymbol{G}^{-}$. The solution obtained at stages 2 and 3 is used as an initial condition (i. e., instead of $\boldsymbol{Q}^n$).

5. The cells of the mesh Ω are traversed in a double loop with respect to j, k in an opposite direction, $j = \overline{N_x - 1, 0}$, $k = \overline{N_y - 1, 0}$. By analogy with stage 3, in each cell, the equations of the chosen scheme are solved for the unknowns $\boldsymbol{Q}$ at the nodes $(x_{j+1/2}, y_{k+1/2})$, $(x_{j+1/2}, y_k)$, $(x_j, y_{k+1/2})$, and (x_j, y_k).
6. The desired solution $\boldsymbol{Q}^{n+1}$ is set equal to the one found at stages 4 and 5, the index n is increased by one, and the process returns to stage 1.

If a high-order DIRK method is used for computations in t, then all its stages are executed at stages 3 and 5. Moreover, in the case of LOD splitting, all stages are first executed for $\boldsymbol{Q}^{[x]}$ (or $\boldsymbol{Q}^{[y]}$) and, then, for $\boldsymbol{Q}^{[y]}$ (or $\boldsymbol{Q}^{[x]}$). In other words, switching between DIRK stages is embedded in switching between the directions of LOD splitting (if any), which is itself embedded in switching between the signs of split fluxes.

The above description shows that the implementation of schemes (2) and (3) is reduced to sequential loops over cells, with the desired solution computed locally in each cell. Beginning the computation of a current cell, we always know the desired solution on two of its faces sharing a vertex, which is guaranteed by the flux splitting procedure and the traversal order of the cells.

Thus, the fact that the spatial stencil of bicompact schemes occupies a single cell, together with their local solvability, makes them very convenient for computations on Cartesian meshes with AMR.

2. Bicompact schemes on Cartesian meshes with solution-based AMR

In what follows, we consider Cartesian meshes with solution-based AMR in a closed domain $\overline{D}$ of simplest geometry. Nevertheless, the constructions and arguments used in this section can be carried over in part to Cartesian meshes with geometry-based AMR (e. g., data structures and cell traversal algorithms).

First, we describe the design of adaptive Cartesian meshes and related data structures. Our consideration will almost completely follow [27].

Assume that a rather coarse standard Cartesian mesh Ω is introduced in $\overline{D}$ (see Section 1). The mesh is made adaptive as follows. Initially or in the course of the computation, each mesh cell can be split (refined) into four new ones by bisecting it in each direction. Each of the four new cells can be split into another four in a similar manner, and so on. The cell to be split is called a parent, while four new cells appearing from it are called children. The number of recursive mesh refinements required for obtaining a cell is called its rank and is denoted by $R \geqslant 0$. For example, the original coarse cells have the rank $R = 0$; applying a single mesh refinement to them yields cells of rank $R = 1$, which are split into cells of rank $R = 2$, and so on. Cell ranks are bounded above: $R \leqslant R_{\max}$. The side lengths of a cell of rank R are expressed in terms of those of the zero-rank cell generating it by the obvious formulas

$$h_x(R) = 2^{-R} h_x(0), \quad h_y(R) = 2^{-R} h_y(0).$$

In the course of computations, the children of a parent cell may merge back in this cell, but zero-rank cells are not allowed to merge. An example of adaptive mesh refinement of a zero-rank cell is shown in Fig. 1 on the left.

To represent two-dimensional adaptive Cartesian meshes, it is useful to use forests of quadtrees (referred to hereinafter as trees for the sake of brevity). Each tree has a zero-rank cell as a root; its children, children of its children, and so on make up the other nodes of this tree. An example of a tree of cells is given in Fig. 1 (right). The edges of the tree are colored in red.

If a cell has no children, it is called computational. Computational cells are leafs in a tree, and it is to them that the numerical scheme is applied. Trees of cells describe the complete history of mesh refinement: they contain not only computational cells, but also all their parents up to the zero rank.

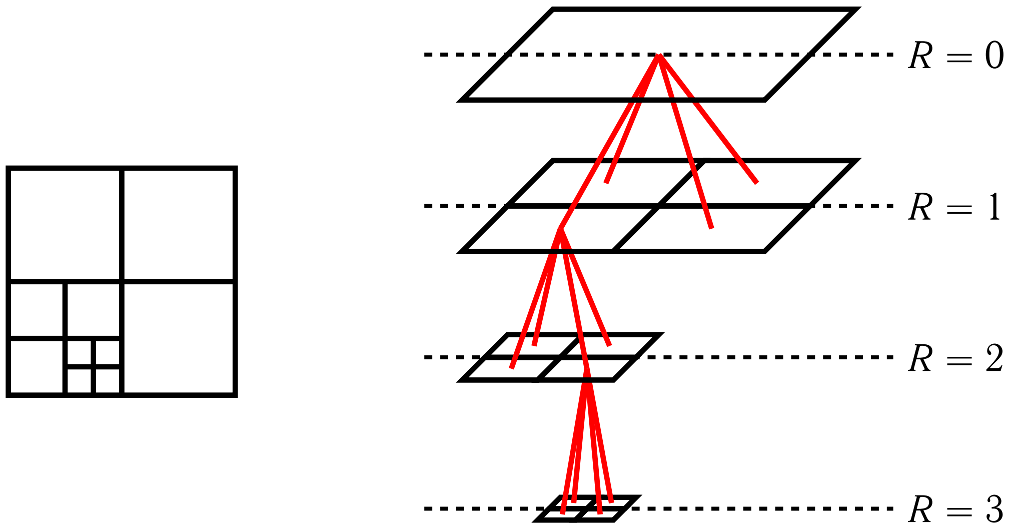

Fig. 1. Adaptive refinement of a zero-rank cell and a tree of cells

An important remark specific to the spatial stencil of bicompact schemes has to be made: the storage of all nine values of the solution $\boldsymbol{Q}$ in a cell that is itself stored in a tree is inefficient, since the data are repeatedly duplicated in this case. For example, even for $R_{\max} = 0$, this approach leads to an almost quadruple increase in the storage requirements for cell vertices and to a double increase for centers of their faces. For $R_{\max} > 0$, the situation is worsened by the fact that all vertices of children cells also belong to their parent cell. A more efficient approach to data storage is as follows.

1. Each cell stores not the solution, but rather the integer coordinates of its nodes (in addition to children pointers and cell parameters, such as the rank, etc.). By the integer coordinates of a point, we mean its coordinates (x, y) measured in terms of the units $h_x(R_{\max})/2$ and $h_y(R_{\max})/2$, respectively.
2. The solution is stored in an associative array, each element of which represents a key-value pair, where the key is the pair of integer coordinates of a mesh node and the value is the solution value at this node. This array is shared by the entire forest of trees. In performing some computations in a cell, the solution values at its nodes are obtained by accessing this array through the integer coordinates of a node. New elements are added to the array in the case of mesh refinement, while some elements are deleted from the array in the case of merging cells.

For this data structure, there are no duplicated solution values, since the number of elements in the associative array is equal precisely to the number of nodes of the mesh Ω. Note that the C++ language involves the standard `map` container, which is well suited for implementing the array described above.

Following [27], as before, we describe the solution-based AMR procedure. Consider the transition from the level t^n to t^{n+1}. Before executing the scheme-based computations, the mesh is adaptively refined using the known solution $\boldsymbol{Q}^n$. The AMR procedure consists of two stages. At the first, all node cells (not only computational ones) of all trees are flagged in some way. At the second stage, the mesh is reformed according to the cell labels.

At the flagging stage, the roots of all trees, i. e., zero-rank cells are traversed in an arbitrary order (for example, in a double loop) and each tree is traversed in postorder. Note that the order of children traversal in recursion is of no importance. In each computational cell, we compute the adaptation criterion

$$d_{s,i} = |\nabla Q_s(x_i, y_i, t^n)| \left(h_{x,i}\, h_{y,i}\right)^{\frac{\omega_0+1}{2\omega_0}}, \tag{6}$$

where$s = \overline{1, m}$ is the index of a component of the vector $\boldsymbol{Q}$, $i = \overline{1, N_c}$ is the index of a computational cell, N_c is the total number of computational cells, $|\boldsymbol{a}|$ is the Euclidean norm of the vector $\boldsymbol{a}$, $\nabla = (\partial_x, \partial_y)$ is the gradient with respect to the space variables, (x_i, y_i) are the coordinates of a cell center, $h_{x,i}$ and $h_{y,i}$ are the cell side lengths in the x and y directions, respectively, ω_0 is a tuned parameter of the AMR algorithm. In the case of bicompact schemes,

$|\nabla Q_s(x_i, y_i, t^n)|$ in formula (6) is approximated by the expression

$$|\nabla Q_s(x_i, y_i, t^n)| \approx \sqrt{\left[A_0^y \Lambda_1^x Q_s^n(x_i, y_i)\right]^2 + \left[\Lambda_1^y A_0^x Q_s^n(x_i, y_i)\right]^2}.$$

The mean square values of $d_{s,i}$ over all computational cells are also obtained in mesh traversal:

$$\sigma_s = \sqrt{\frac{1}{N_c} \sum_{i=1}^{N_c} d_{s,i}^2}.$$

After all $d_{s,i}$ and their mean square values σ_s were computed, the forest of trees is traversed once again in the same order. Each tree node and its cell are assigned one of the following three possible flags.

1. If a cell is computational (without children), its rank is $R_i < R_{\max}$, and there exists an index $s = \overline{1,m}$ such that $d_{s,i} \geqslant \omega_1 \sigma_s$, then this cell is flagged for refinement.
2. If a cell is computational (without children), its rank is $R_i > 0$, and the inequality $d_{s,i} \leqslant \omega_2 \sigma_s$ holds for all $s = \overline{1,m}$, then this cell is flagged for coarsening.
3. In the other cases, the cell is flagged for “nothing to do”.

Here, ω_1 and ω_2 are also tuned parameters of the AMR algorithm. The parameters $\omega_0, \omega_1, \omega_2$ have the following ranges of variation:

$$\omega_0 > 0, \quad \omega_1 \in [1.0, 1.5], \quad \omega_2 \in [0.1, 0.5].$$

In [27], they were specified as

$$\omega_0 = 2, \quad \omega_1 = 1.0, \quad \omega_2 = 0.1.$$

These parameters can be interpreted as follows: ω_0 is the weight with which the cell’s length scale $\sqrt{h_{x,i}\, h_{y,i}}$ is involved in adaptation criterion (6), ω_1 determines how “easily” cells are refined (refinement is more likely for smaller values of ω_1), and ω_2, on the contrary, determines how “easily” cells are coarsened (coarsening is more likely for larger values of ω_2).

Note that criterion (6) differs substantially from those that were used in [27] and more recent works based on [27]. The criterion of [27] and similar ones available in the literature are intended for fluid mechanics and are not suitable for other applications, while criterion (6) is formulated for a general hyperbolic system. Moreover, the need for adaptive refinement in available criteria is estimated in terms of either the density gradient, or pressure gradient, or velocity divergence, or velocity curl, or a combination of the last two quantities. The shortcomings of these variants were discussed in [27], and a combination of the divergence and the curl of velocity was chosen as the most successful one, though incapable of recognizing plane contact discontinuities. Another shortcoming of the combined criterion is that it ignores any perturbations against the background of zero or constant velocity. The new criterion (6) proposed in this work takes into account the variations in all solution components and yields close-to-zero values of $d_{s,i}$ for all $s = \overline{1, m}$ only in a cell where the solution is nearly identically constant in all components.

At the remeshing stage, the roots of all trees are traversed as before in an arbitrary order, each tree is traversed in preorder, while the order of children traversal in recursion is of no importance. The tree nodes and their cells are transformed according to flags given before.

1. If a cell is flagged for refinement, then it is divided into four new cells by bisecting it in each direction. These four new cells are added to the tree as children of the divided cell. The mesh is supplemented with eight to sixteen new nodes depending on the degree to which the neighbors of the divided cell are refined (some candidates for new nodes may already exist), and the same number of new elements are added to the associative array storing the solution. The solution at the new nodes is computed using biquadratic interpolation [28, Eq. (9)] constructed for the divided cell. Each of the four new cells is flagged for “nothing to do”.
2. If each of the four children of a parent cell is flagged for coarsening, then they are merged back into their parent and are deleted from the tree. The nodes of the merged cells are deleted from the mesh only if they do not belong to other cells (the centers of merged cells are always deleted). The same number of elements corresponding to deleted nodes are deleted from the associative array.
3. In the other cases, nothing happens to a cell.

At the zero time level t^0, the solution-based AMR procedure is run $R_{\max}$ times, so that the mesh is maximally adapted to the initial condition before the beginning of the scheme computations. If the initial condition is specified by a given function of coordinates x, y, rather than by tabulated values, then, in the case of mesh refinement, it is reasonable to compute the solution at new nodes by projecting the initial condition onto the mesh, rather than by applying interpolation.

After the mesh was adapted, the solution $\boldsymbol{Q}^{n+1}$ at the next time level is computed using the chosen scheme. Note that the next mesh refinement is performed after finding $\boldsymbol{Q}^{n+1}$, i. e., the mesh remains unchanged in advancing one time step.

Except for several details, the algorithm implementing bicompact schemes on Cartesian meshes with solution-based AMR is nearly the same as in the case of standard Cartesian meshes (see the description of the latter at the end of Section 1).

One difference from standard Cartesian meshes is that computational cells of adaptive Cartesian meshes are rather difficult to index and traverse in a "standard" nonrecursive loop in proper order. It is much simpler to traverse computational cells via the forest of trees. Let an arbitrary cell/tree node be denoted by P and its children cells (if any) be denoted by LL, LR, RL, and RR (see Fig. 2, where P is depicted by a dashed line). The roots of all trees, i. e., zero-rank cells are traversed in a double loop in the direction corresponding to the signs of the split flux vectors by analogy with cells of a standard Cartesian mesh. Each of the trees is traversed in preorder: if the cell P is not computational, then its children are traversed; if P is computational, then the bicompact scheme is implemented there. Clearly, the order of children traversal depends on the signs of the split flux vectors. The possible variants are

$$\begin{aligned} &\boldsymbol{F}^+, \boldsymbol{G}^+ \Rightarrow \text{LL, LR, RL, RR}; \quad \boldsymbol{F}^-, \boldsymbol{G}^- \Rightarrow \text{RR, RL, LR, LL}; \\ &\boldsymbol{F}^+, \boldsymbol{G}^- \Rightarrow \text{LR, LL, RR, RL}; \quad \boldsymbol{F}^-, \boldsymbol{G}^+ \Rightarrow \text{RL, RR, LL, LR}. \end{aligned} \tag{7}$$

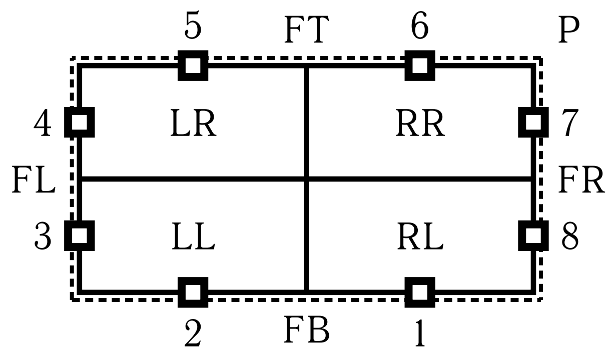

Fig. 2. Notation for an adaptive Cartesian mesh

Another difference is that the following situation can happen: in a certain computational cell, the sought solution required at some nodes of the cell faces is still unknown, although all preceding computational cells (in the order of traversal) have been processed. This occurs in marching in space when the bicompact scheme proceeds from large to small computational cells. For the fluxes $\boldsymbol{F}^+, \boldsymbol{G}^+$, this situation is illustrated in Fig. 3. In computing cells 1–3, the desired solution is found at the nodes marked with open green boxes in Fig. 3. Next, the traversal algorithm processes the large cell in the upper right corner, which is split into computational cells 4–7. Trying to compute cells 4–6, we see that the desired solution at the nodes marked with red boxes is still unknown (without this solution, cells 4–6 cannot be computed). However, this difficulty can easily be overcome if we compute the unknown solution values in terms of known ones by applying one-dimensional quadratic interpolation similar to [28, Eq. (3)] on the left and lower faces of the large cell made up of cells 4–7.

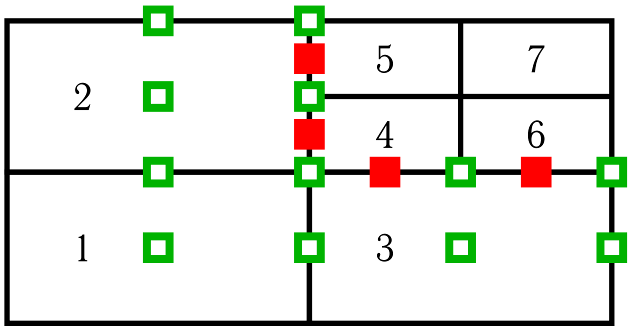

Fig. 3. Shortage of data in passing from large to small cells

In the general form, the extra procedure for computing the desired solution by interpolation on faces can be described as follows. Let the faces of an arbitrary node cell P be denoted by FB, FL, FT, and FR (see Fig. 2, where potentially problematic nodes are shown by open markers indexed by 1 to 8). The extra computation procedure is built in the traversal over the forest of trees in which the bicompact scheme is implemented.

1. If the cell P is computational, then compute it according to the scheme.
2. If the cell P is not computational, then, before passing to its children LL, LR, RL, RR, check the following nodes:

$$1,\ 2,\ 3,\ 4 \text{ for fluxes } \boldsymbol{F}^+, \boldsymbol{G}^+; \quad 5,\ 6,\ 7,\ 8 \text{ for fluxes } \boldsymbol{F}^-, \boldsymbol{G}^-;$$
$$3,\ 4,\ 5,\ 6 \text{ for fluxes } \boldsymbol{F}^+, \boldsymbol{G}^-; \quad 7,\ 8,\ 1,\ 2 \text{ for fluxes } \boldsymbol{F}^-, \boldsymbol{G}^+.$$

   If the desired solution is still unknown at nodes

$$1,\ 2 \Rightarrow \text{compute using 1D interpolation on the face FB;}$$
$$3,\ 4 \Rightarrow \text{compute using 1D interpolation on the face FL;}$$
$$5,\ 6 \Rightarrow \text{compute using 1D interpolation on the face FT;}$$
$$7,\ 8 \Rightarrow \text{compute using 1D interpolation on the face FR.}$$

   Next, the children of P are traversed in the order described in (7).

Since one-dimensional quadratic interpolation on any cell face coincides with two-dimensional biquadratic interpolation over the entire cell taken on this face, the extra computation of the desired solution on it does not violate the bicompact approximation.

Thus, we have completely described the implementation of bicompact schemes on Cartesian meshes with solution-based AMR in the simplest computational domains.

3. Numerical experiments

Let us test bicompact schemes on Cartesian meshes with solution-based AMR on a pair of two-dimensional benchmark problems. In all the computations, the AMR parameters are specified as $\omega_0 = 2$, $\omega_1 = 1.0$, and $\omega_2 = 0.1$.

Advection of a compactly supported pulse. This problem is stated as follows. We solve the simplest version of system (1), namely, the linear advection equation

$$\mathscr{L}_2(u) \equiv \partial_t u + a\,\partial_x u + b\,\partial_y u = 0, \quad a = \text{const} > 0, \quad b = \text{const} > 0,$$

where $u = u(x, y, t)$ is the sought function. The advection velocities a, b are set equal to 1. The computational domain is $D = (0, 1) \times (0, 1)$ ($x_{\max} = y_{\max} = 1$), and the maximum time is $t_{\max} = 0.5$. The initial condition is specified as

$$u(x, y, 0) = P_7^f\left[4\sqrt{(x - 1/4)^2 + (y - 1/4)^2}\right], \quad (x, y) \in \overline{D},$$

where $P_q^f(x) \in C^{q-1}(\mathbb{R})$, $q \in \mathbb{N}$, is a compactly supported polynomial of adjustable smoothness:

$$P_q^f(x) = \begin{cases} (1 - x^2)^q & \text{if } |x| < 1, \\ 0 & \text{if } |x| \geqslant 1. \end{cases}$$

The boundary values are constant (zero), and they are set on the boundaries $x = 0$ and $y = 0$ for all $t \in (0, t_{\max}]$.

The problem was computed using two bicompact schemes of fourth-order accuracy in x, y, namely, T2B4 and SDIRK3B4. Time stepping in the former scheme was based on the second-order trapezoidal rule. In the latter scheme, we applied the L-stable stiffly accurate three-stage SDIRK method of third order [29, Eq. (17)]. No flux splitting was used in this problem, since the advection velocities were always positive.

We begin with quality verification. To determine whether the advection of the pulse and the mesh adaption are correctly reproduced, we consider the results generated by T2B4 in a single run. The mesh and scheme parameters were specified as

$$h_x(0) = h_y(0) = 0.1, \quad R_{\max} = 3, \quad h_x(R_{\max}) = h_y(R_{\max}) = 0.0125,$$
$$\tau = 0.005, \quad \kappa_x(0) = \kappa_y(0) = 0.05, \quad \kappa_x(R_{\max}) = \kappa_y(R_{\max}) = 0.4,$$

where $\kappa_x = a\tau/h_x$ and $\kappa_y = b\tau/h_y$ are the Courant numbers in x, y, respectively.

Figs. 4–6 present two-dimensional plots of the solution (in color) and computational cells of the corresponding mesh at the times $t = 0, 0.25, 0.5$. It can be seen that T2B4 correctly reproduces the advection of the pulse (without amplitude loss). The mesh is adapted adequately, its refinement follows the pulse, and, in constant-solution areas, the cells merge into ones of zero rank.

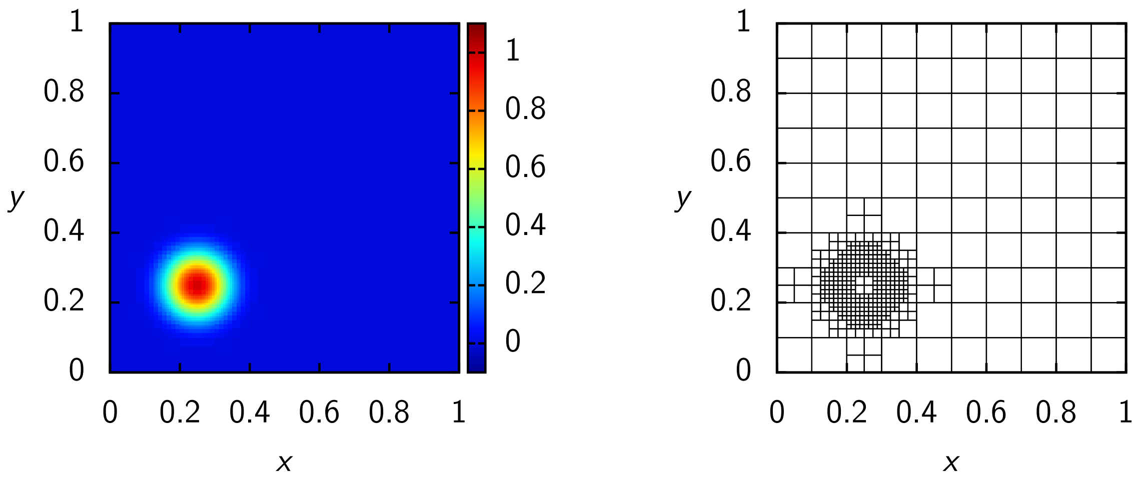

Fig. 4. Solution and mesh at $t = 0$

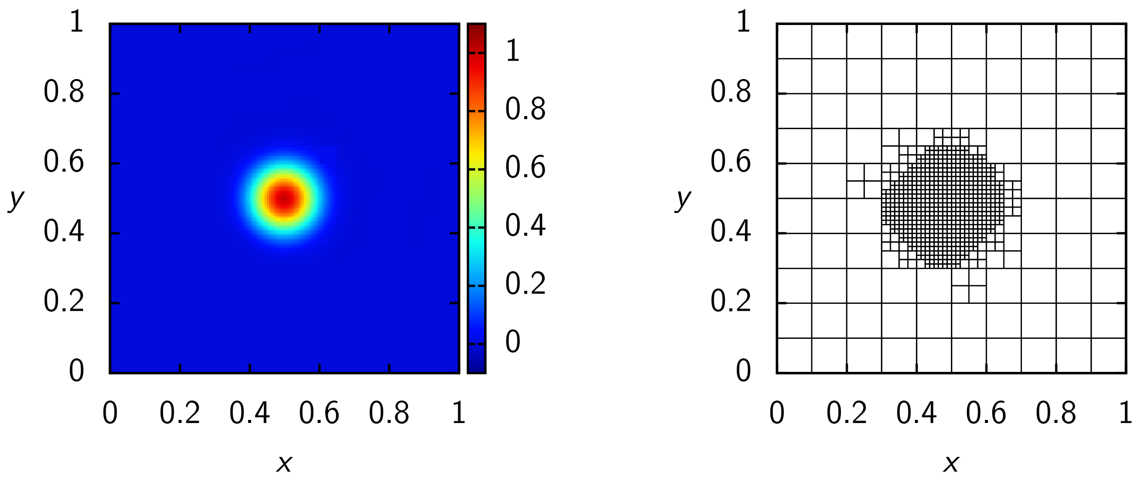

Fig. 5. Solution and mesh at $t = 0.25$

Fig. 7 shows the square root of the number of computational cells as a function of time. The curve $\sqrt{N_c(t)}$ quickly reaches a constant level, which agrees with the fact that the pulse is compactly supported. This plot suggests that, in terms of the number of computational cells, the computational complexity of the scheme on an adaptive mesh with the above-indicated parameters and identical τ is equivalent to that on a uniform 27×27 mesh.

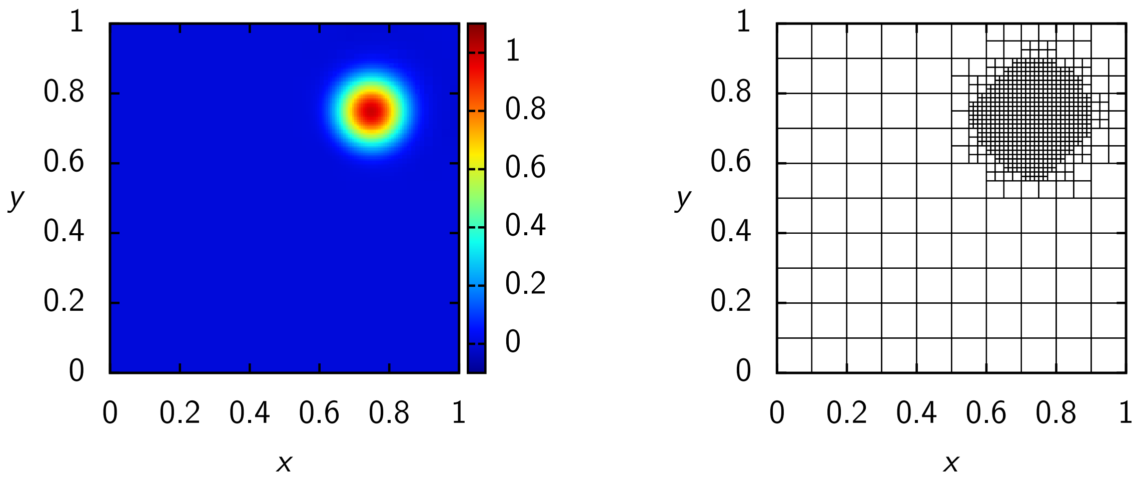

Fig. 6. Solution and mesh at $t = 0.5$

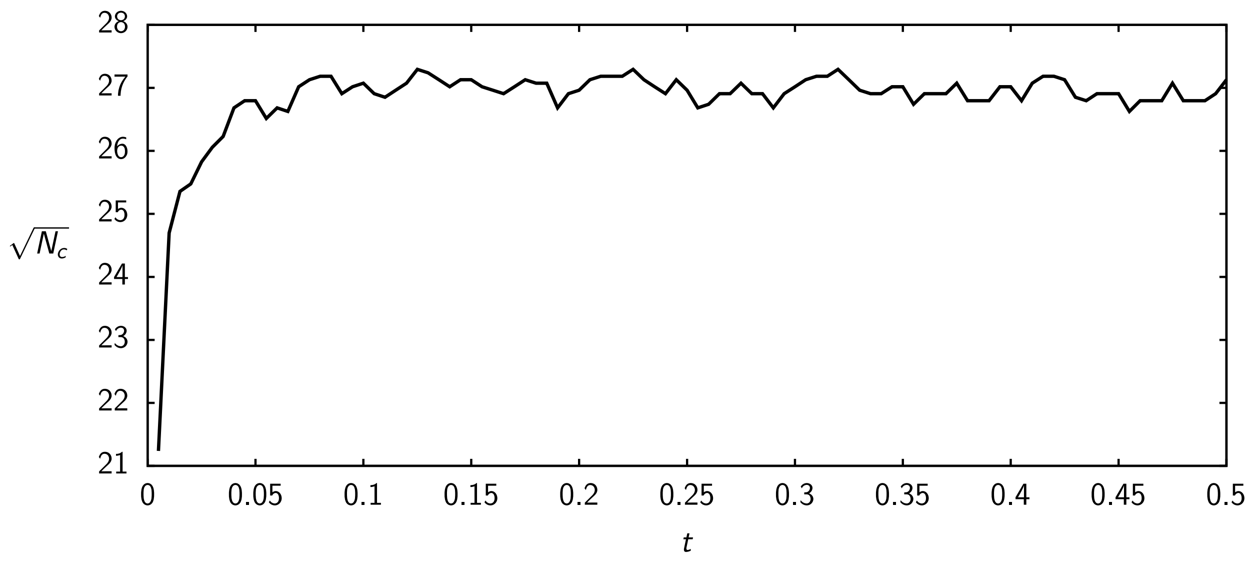

Fig. 7. Square root of the number of computational cells as a function of time

Let us analyze the mesh convergence of SDIRK3B4 for the exact solution of the problem under consideration. The mesh is refined as follows:

$$h_x(0) = h_y(0) = h = 0.1, 0.05, 0.025, 0.0125, \quad R_{\max} = \text{const},$$
$$\kappa_x(R_{\max}) = \kappa_y(R_{\max}) = \kappa = \text{const}, \quad \tau = \frac{h}{2^{R_{\max}}\kappa}.$$

The refinement is performed for four values $R_{\max} = 0, 1, 2, 3$ and three values $\kappa = 1.0, 0.5, 0.2$ (altogether 48 runs). We are interested in E_∞ — the absolute error of the scheme in the L_∞ norm computed at $t = 0.5$ over all mesh nodes. Since the number of computational cells varies with time, it is reasonable to show E_∞ as a function of the total number N_{sum} of cells over all levels:

$$N_{\text{sum}} = \sum_{n=0}^{N_t} N_c(t^n).$$

Another advantage of N_{sum} is that its values can be used to compare the computational complexity of the scheme on different meshes, both uniform and adaptive.

Fig. 8 shows the plots of E_∞ against N_{sum} for various $R_{\max}$ and $\varkappa$. These results suggest that SDIRK3B4 on an adaptive mesh with $R_{\max} = 1, 2$ and not too small $\varkappa = 1.0, 0.5$ has an a posteriori order of convergence close to the theoretical one. For $R_{\max} = 3$, the order of convergence is somewhat lower than the theoretical one, which is quite noticeable for $\varkappa = 0.2$. Note that, for $\varkappa = 1.0, 0.5$, due to the AMR, the prescribed error in the solution is achieved with a smaller N_{sum}, i. e., at a lower computational complexity. For small Courant numbers, however, the solution-based AMR hardly contributes to the speed-up of the computation.

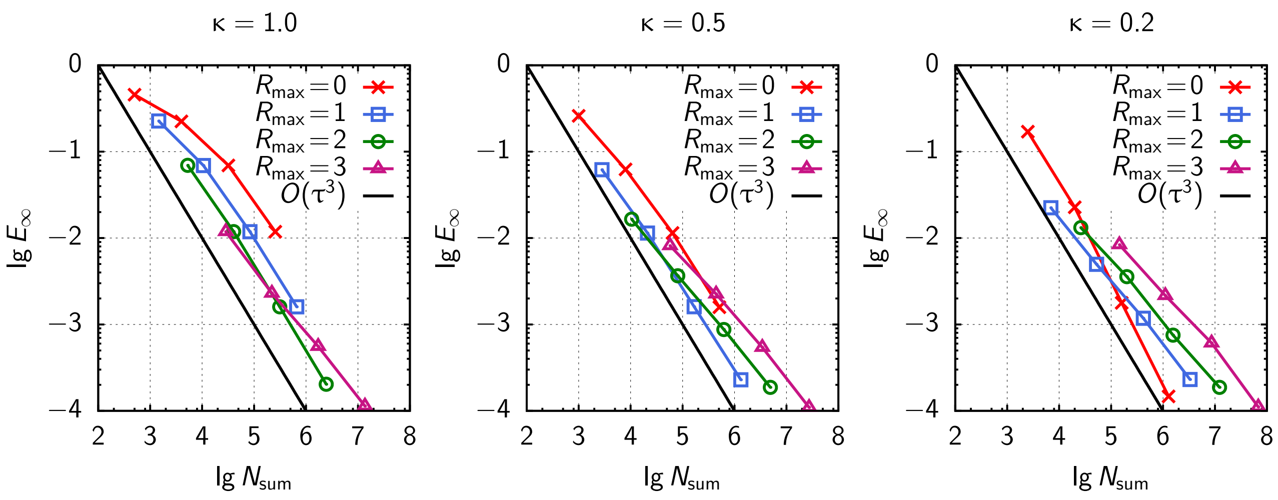

Fig. 8. Absolute errors in the L_∞ norm against the total number of computational cells for various $R_{\max}$ and $\varkappa$

Sedov blast wave problem. The solution of this well-known gasdynamic problem [30] consists of a strong shock wave, a nearly constant flow field in a neighborhood of the blast point behind the shock wave, and a constant background in the outer region. This solution structure is well suitable for testing schemes on meshes with solution-based AMR.

The system of two-dimensional gasdynamic Euler equations has the form of (1), where

$$\boldsymbol{Q} = \begin{bmatrix} \rho \\ \rho v_x \\ \rho v_y \\ E \end{bmatrix}, \quad \boldsymbol{F}(\boldsymbol{Q}) = \begin{bmatrix} \rho v_x \\ \rho v_x^2 + p \\ \rho v_x v_y \\ v_x(E+p) \end{bmatrix}, \quad \boldsymbol{G}(\boldsymbol{Q}) = \begin{bmatrix} \rho v_y \\ \rho v_x v_y \\ \rho v_y^2 + p \\ v_y(E+p) \end{bmatrix},$$

$$E = \frac{p}{\gamma - 1} + \frac{\rho|\boldsymbol{v}|^2}{2}.$$

Here, ρ, $\boldsymbol{v} = (v_x, v_y)$, p, and E denote the density, velocity, pressure, and specific total energy (per unit volume), respectively, and $\gamma = 1.4$ is the ratio of specific heats (the gas is diatomic). The computational domain is $D = (0,2) \times (0,2)$ ($x_{\max} = y_{\max} = 2$), and the maximum time is $t_{\max} = 0.01$.

The initial and boundary conditions are set as follows. The blast occurs at the time $t = 0$ at the point (x_0, y_0) with $x_0 = y_0 = 1$. The blast energy $\mathscr{E}_0$ is specified so that the radius of the cylindrical shock wave is 0.8 at the time $t = t_{\max} = 0.01$; namely, $\mathscr{E}_0 = 4030.78$. The initial conditions are as follow: for $(x,y) \in \Omega$,

$$\rho(x,y,0) = 1, \quad v_x(x,y,0) = v_y(x,y,0) = 0,$$
$$E(x,y,0) = \begin{cases} \mathscr{E}_0/[h_x(R_{\max})h_y(R_{\max})] & \text{if } (x,y) = (x_0,y_0), \\ 10^{-2}/(\gamma-1) & \text{otherwise.} \end{cases} \tag{8}$$

The steps $h_x(0)$ and $h_y(0)$ are chosen so that $(x_0, y_0) \in \Omega$. Before starting the run in t, the mesh is adapted to initial conditions (8) $R_{\max}$ times, after which the values of $\boldsymbol{Q}^0$ at half-integer nodes are updated using its values at integer nodes by applying bilinear interpolation (i. e., by averaging). The boundary conditions remain unchanged, and they are set on the entire ∂D.

This problem was computed using the bicompact scheme SDIRK3B4 with symmetrized LOD splitting and a conservative limiting method [28]. Following the approach of [28], SDIRK3B4 is scheme B; as a scheme A, we used a baseline bicompact scheme with unsymmetrized LOD splitting.

The parameters of the scheme were $C_1 = 0.5$, $\sigma = 0.1$ ([28]); C_2^x and C_2^y were automatically determined by formulas (5) with $\delta = 0.2$. The time step τ was variable and was computed using the formula

$$\tau = 2\kappa \min\left\{ \frac{h_x(R_{\max})}{V_{\max}^x + 2C_2^x}, \frac{h_y(R_{\max})}{V_{\max}^y + 2C_2^y} \right\},$$

where κ is the Courant number in cells of maximum rank. We specified $\kappa = 0.8$. The nonlinear equations of schemes A and B were solved using Newton's method with relative error rtol $= 10^{-7}$. The computations were performed on a mesh with $h_x(0) = h_y(0) = 2/160$ and $R_{\max} = 1$.

Fig. 9 shows the two-dimensional plots of density (in color) and the ranks of cells at a finite time. These results suggest that the bicompact LOD scheme SDIRK3B4 with a conservative limiting method yields the correct symmetry of the solution and the shock front is free of carbuncles or any other nonphysical features. Moreover, the bicompact scheme demonstrates stable performance without any special correction procedures of negative densities and pressures. The AMR algorithm performs well: the cells refine in areas where the solution gradient is high and merge in areas where the solution varies little.

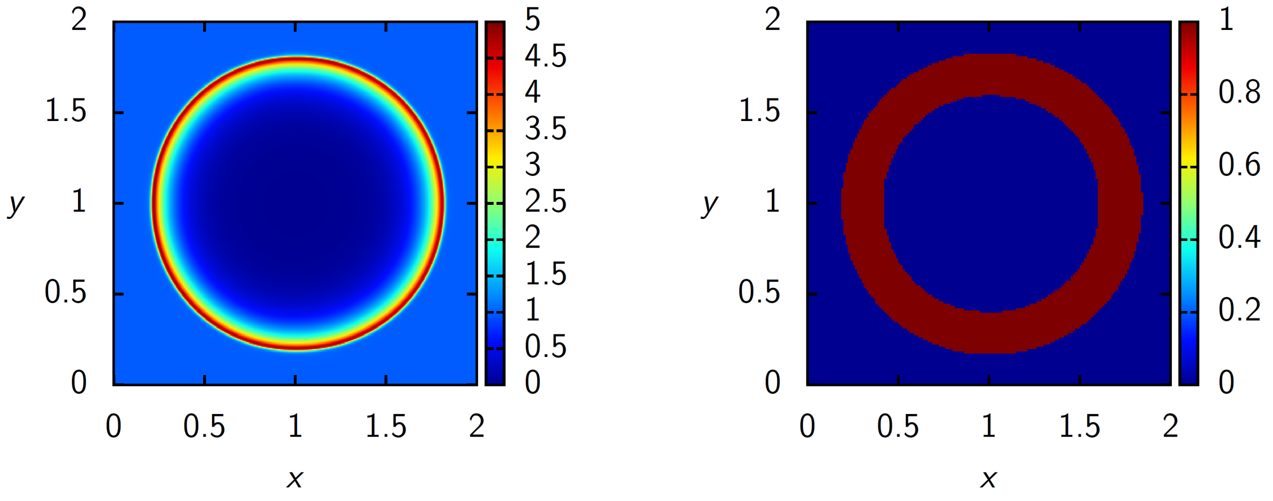

Fig. 9. Density field and ranks of cells at $t = t_{\max} = 0.01$

Figs. 10–12 present the one-dimensional density, velocity, and pressure profiles, respectively, on the interval $x = 1$, $y \in [1, 2]$ at the final time. The numerical solution at integer nodes is shown by color markers, while the exact solution is depicted by the solid curve. An analysis of these profiles suggests that the numerical solution agrees well with the exact one and the computed shock wave spreads only over three cells of zero rank. Moreover, the sharp peaks of density and pressure immediately behind the shock wave are fairly well resolved by the tested scheme.

Fig. 13 shows the square root of the number of computational cells as a function of time. The plot suggests that, in the case of AMR, the number of computational cells in this problem for $R_{\max} = 1$ is halved as compared with a uniform mesh with steps $h_x(R_{\max})$, $h_y(R_{\max})$.

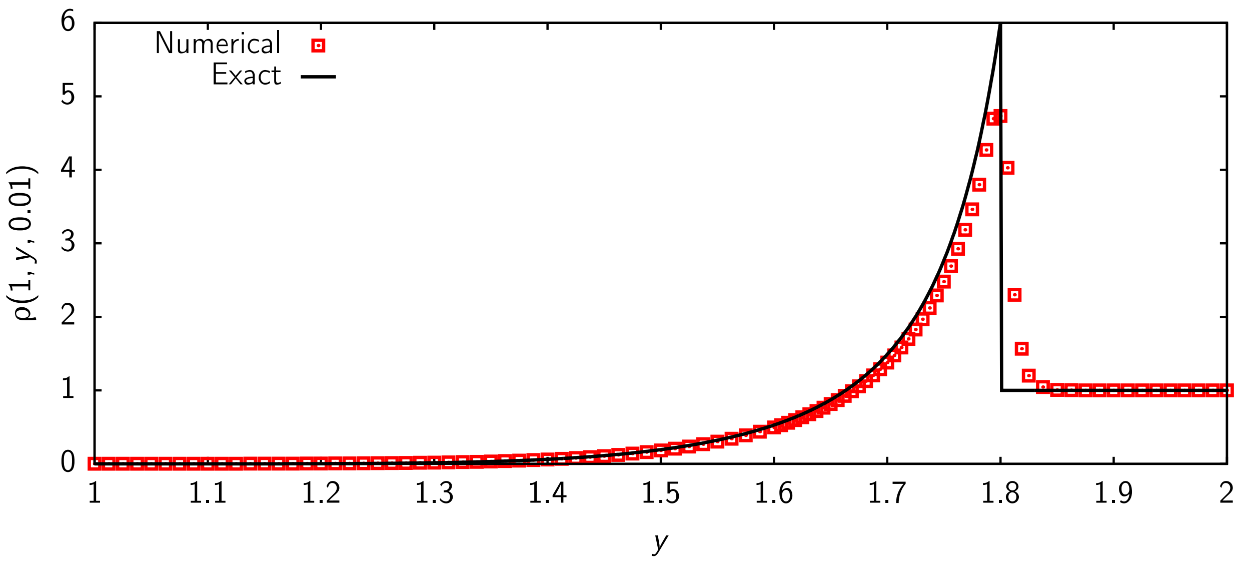

Fig. 10. Density profiles on the interval $x = 1$, $y \in [1, 2]$ at $t = t_{\max} = 0.01$

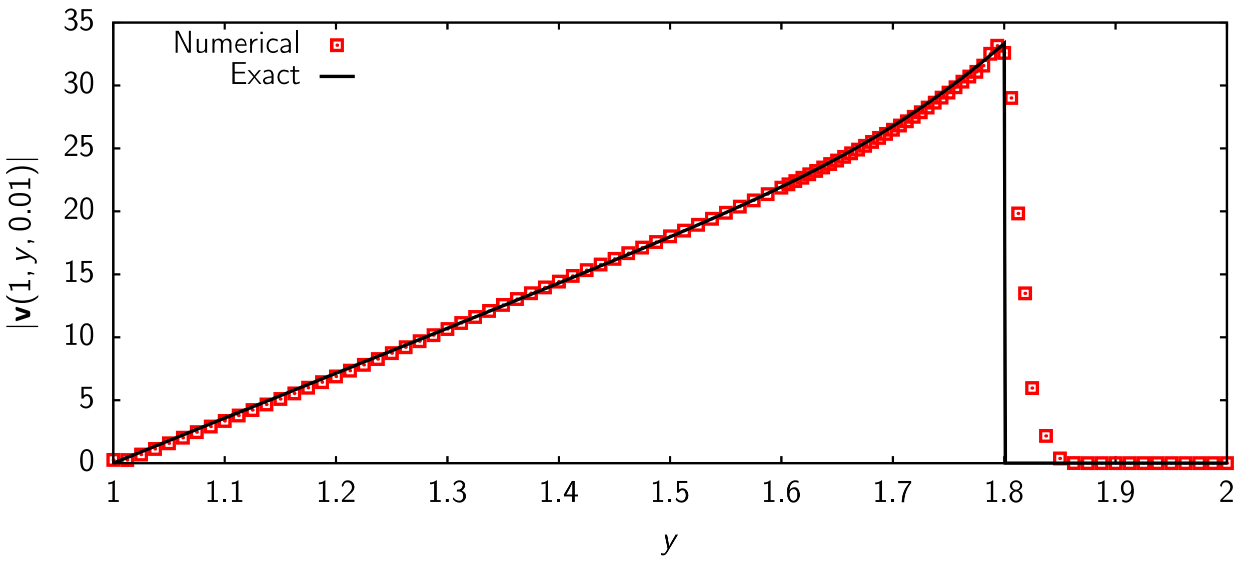

Fig. 11. Velocity profiles on the interval $x = 1$, $y \in [1, 2]$ at $t = t_{\max} = 0.01$

Conclusions

A numerical algorithm based on bicompact schemes of fourth order accuracy in space on Cartesian meshes with solution-based AMR was described. Practical recommendations were given concerning the implementation of data structures for storing the mesh and the mesh function that take into account the specific features of the spatial stencil of bicompact schemes. A new solution-based AMR criterion suitable for general hyperbolic systems of equations was proposed.

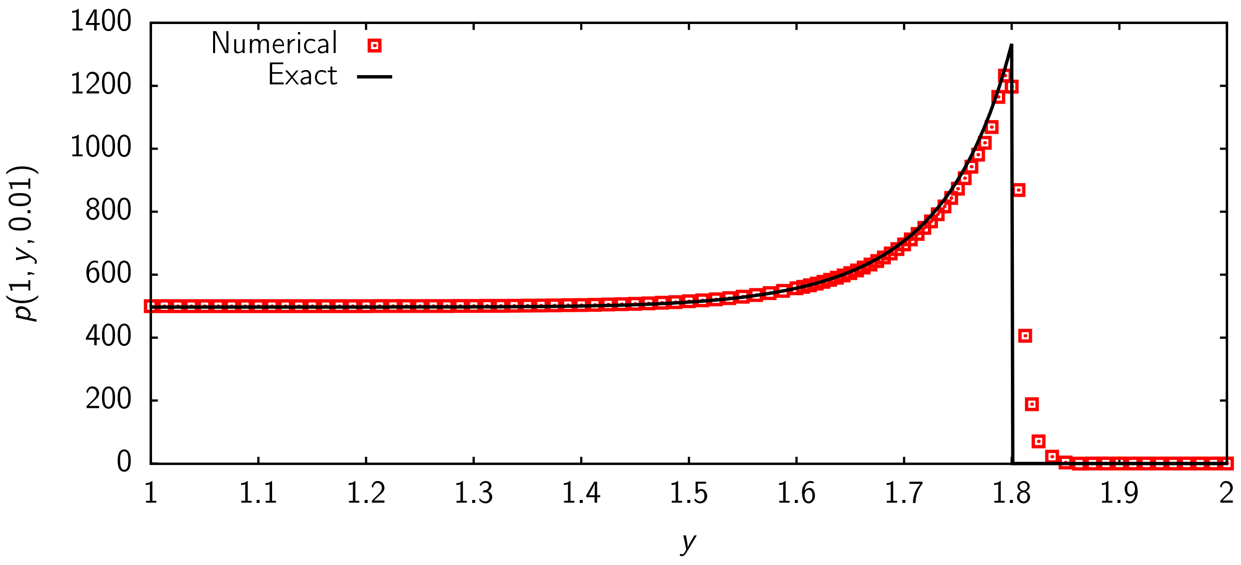

Fig. 12. Pressure profiles on the interval $x = 1$, $y \in [1, 2]$ at $t = t_{\max} = 0.01$

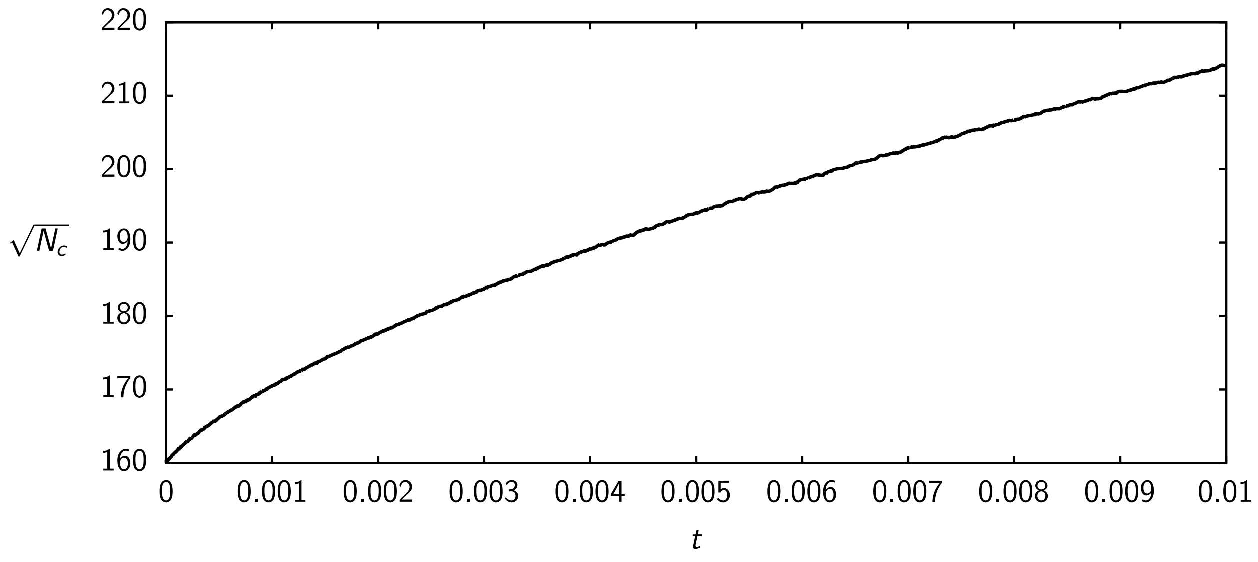

Fig. 13. Square root of the number of computational cells as a function of time

Bicompact schemes on Cartesian meshes with solution-based AMR were used to compute the two-dimensional advection of a compactly supported pulse and the two-dimensional Sedov blast wave problem. An analysis of the results obtained for the first problem showed that bicompact schemes preserve the high order of accuracy on adaptive Cartesian meshes. These schemes demonstrated good accuracy in both problems. Moreover, in the Sedov problem, a bicompact scheme with a conservative limiting method does not require special techniques for correcting negative density and pressure and yields a symmetric solution free of nonphysical features, such as carbuncles. The results of this work lay the foundations for implementing bicompact schemes on Cartesian meshes with geometry-based AMR in computational domains of complex geometry.

Bibliography list

1. Ekaterinaris J. A. High-order accurate, low numerical diffusion methods for aerodynamics // Prog. Aerosp. Sci. — 2005. — Vol. 41. — P. 192–300.
2. Kurbatskii K. A., Mankbadi R. R. Review of computational aeroacoustics algorithms // Int. J. Comput. Fluid Dyn. — 2004. — Vol. 18, no. 6. — P. 533–546.
3. Dumbser M., Käser M., de la Puente J. Arbitrary high-order finite volume schemes for seismic wave propagation on unstructured meshes in 2D and 3D // Geophys. J. Int. — 2007. — Vol. 171. — P. 665–694.
4. A high-order discontinuous Galerkin method for wave propagation through coupled elastic-acoustic media / Lucas C. Wilcox, Georg Stadler, Carsten Burstedde, Omar Ghattas // J. Comput. Phys. — 2010. — Vol. 229. — P. 9373–9396.
5. Hesthaven J. S. High-order accurate methods in time-domain computational electromagnetics: a review // Adv. Imag. Elect. Phys. — 2003. — Vol. 127. — P. 59–123.
6. Mikhailovskaya M. N., Rogov B. V. Monotone compact running schemes for systems of hyperbolic equations // Comput. Math. Math. Phys. — 2012. — Vol. 52, no. 4. — P. 578–600.
7. Rogov B. V. High-order accurate monotone compact running scheme for multidimensional hyperbolic equations // Comput. Math. Math. Phys. — 2013. — Vol. 53, no. 2. — P. 205–214.
8. Chikitkin A. V., Rogov B. V., Utyuzhnikov S. V. High-order accurate monotone compact running scheme for multidimensional hyperbolic equations // Appl. Numer. Math. — 2015. — Vol. 93. — P. 150–163.
9. Bragin M. D., Rogov B. V. Minimal dissipation hybrid bicompact schemes for hyperbolic equations // Comput. Math. Math. Phys. — 2016. — Vol. 56, no. 6. — P. 947–961.

10. Chikitkin A. V., Rogov B. V. Family of central bicompact schemes with spectral resolution property for hyperbolic equations // Appl. Numer. Math. — 2019. — Vol. 142. — P. 151–170.

11. Rogov B. V. Dispersive and dissipative properties of the fully discrete bicompact schemes of the fourth order of spatial approximation for hyperbolic equations // Appl. Numer. Math. — 2019. — Vol. 139. — P. 136–155.

12. Chikitkin A. V., Rogov B. V., Aristova E. N. High-order accurate bicompact schemes for solving the multidimensional inhomogeneous transport equation and their efficient parallel implementation // Dokl. Math. — 2016. — Vol. 94, no. 2. — P. 517–522.

13. Berger M., Colella P. Local adaptive mesh refinement for shock hydrodynamics // J. Comput. Phys. — 1989. — Vol. 82. — P. 64–84.

14. De Zeeuw D., Powell K. An adaptively refined Cartesian mesh solver for the Euler equations // J. Comput. Phys. — 1993. — Vol. 104. — P. 56–68.

15. Hartmann D., Meinke M., Schröder W. An adaptive multilevel multigrid formulation for Cartesian hierarchical grid methods // Comput. Fluids. — 2008. — Vol. 37. — P. 1103–1125.

16. Ji H., Lien F.-S., Yee E. A robust and efficient hybrid cut-cell/ghost-cell method with adaptive mesh refinement for moving boundaries on irregular domains // Comput. Methods Appl. Mech. Engrg. — 2008. — Vol. 198. — P. 432–448.

17. Park S., Shin H. Efficient generation of adaptive Cartesian mesh for computational fluid dynamics using GPU // Int. J. Numer. Meth. Fluids. — 2012. — Vol. 70, no. 11. — P. 1393–1404.

18. A Cartesian grid embedded boundary method for the compressible Navier–Stokes equations / Daniel T. Graves, Phillip Colella, David Modiano et al. // Comm. App. Math. and Comp. Sci. — 2013. — Vol. 8, no. 1. — P. 99–122.

19. Positivity-preserving Runge-Kutta discontinuous Galerkin method on adaptive Cartesian grid for strong moving shock / Jianming Liu, Jianxian Qiu, Mikhail Goman et al. // Numer. Math. Theor. Meth. Appl. — 2016. — Vol. 9, no. 1. — P. 87–110.

20. Buchmüller P., Dreher J., Helzel C. Finite volume WENO methods for hyperbolic conservation laws on Cartesian grids with adaptive mesh refinement // Appl. Math. Comput. — 2016. — Vol. 272, no. 2. — P. 460–478.

21. Adaptive wavelet algorithms for solving problems of hydro- and gas dynamics on Cartesian grids / A. L. Afendikov, A. A. Davydov, A. E. Lutsky et al. — Moscow : Keldysh Institute of Applied Mathematics, 2016. — 232 p. — (In Russian).

22. Marchuk G. I. Splitting methods. — Moscow : Nauka, 1988. — 263 p. — (In Russian).

23. Samarskii A. A. The theory of difference schemes. — New York : Marcel Dekker, 2001. — 762 p.

24. Yanenko N. N. The method of fractional steps: the solution of problems of mathematical physics in several variables. — Berlin : Springer-Verlag, 1971. — 160 p.

25. Bragin M. D., Rogov B. V. On exact dimensional splitting for a multidimensional scalar quasilinear hyperbolic conservation law // Dokl. Math. — 2016. — Vol. 94, no. 1. — P. 382–386.

26. Bragin M. D., Rogov B. V. Iterative approximate factorization of difference operators of high-order accurate bicompact schemes for multidimensional nonhomogeneous quasilinear hyperbolic systems // Comput. Math. Math. Phys. — 2018. — Vol. 58, no. 3. — P. 295–306.

27. De Zeeuw D. A quadtree-based adaptively-refined Cartesian-grid algorithm for solution of the Euler equations. PhD Thesis. — Ann Arbor : The University of Michigan, 1993. — 149 p.

28. Bragin M. D., Rogov B. V. A conservative limiting method for bicompact schemes // Keldysh Institute Preprints. — 2019. — no. 8. — 25 p.

29. Skvortsov L. M. Diagonally implicit Runge–Kutta FSAL methods for stiff and differential-algebraic systems // Mat. Model. — 2002. — Vol. 14, no. 2. — P. 3–17.

30. Sedov L. I. Similarity and dimensional methods in mechanics. — 10th edition. — Boca Raton, Florida : CRC Press, 1993.

Contents